Towards an Ontological Representation of Services in Search Computing

Fabian Suchanek, Alessandro Bozzon, Emanuele Della Valle, Alessandro Campi, Stefania Ronchi

Towards an Ontological Representation of Services in Search Computing

F. Suchanek[1], A. Bozzon[2], E. Della Valle[2], A. Campi[2], and S. Ronchi[2]

[1] INRIA Saclay, Paris, France**
[2] Politecnico di Milano, Dipartimento di Elettronica e Informazione, P.za L. Da Vinci, 32. I-20133 Milano - Italy

Abstract. In the Search Computing project, Web services are modeled by the Semantic Resource Framework (SRF). In this article, we argue that the SRF could benefit from ontological concepts borrowed from the Semantic Web. We first present the knowledge representation used in the Semantic Web, notably in the YAGO ontology [14]. We show how this model is used in the ANGIE system [12] to represent Web Services in conjunction with YAGO. We draw parallels to the Service Mart [3] model used in SeCo. We propose a symbiosis of the two models, discussing the challenges and advantages that come with the integrated model.

1 Introduction

The Search Computing project (SeCo) [3] uses the Semantic Resource Framework (SRF) to model Web services. The SRF is a multi-layer model. The higher layers provide an abstract semantic description of the services, building on the notions of *Service Marts* and *Connection Patterns*. The lower layers (*service interfaces* and *access patterns*) are concerned with the physical properties of the services. Ideally, every service belongs conceptually to a Service Mart. A Service Mart is structurally defined by means of attributes. Two Service Marts can be connected by a *Connection Pattern*. At the logical level, each Service Mart is associated with one or more access patterns representing the signatures of the service calls. *Access patterns* contain a subset of the attributes of the Service Mart, which are tagged with either I (input), or O (output). Attributes can also be tagged as R (ranking), to denote attributes that are used for ordering result instances. Ranking is particularly important in SeCo, because it allows mastering the combinatory explosion of multi-domain queries typical in Search Computing.

By design, the creation of a SRF is a bottom-up process, whereby the real world entities modeled by the Service Marts are typically created on the basis of the Web services. In this article, we try to anticipate how SeCo can cope with a large scale deployment scenario with a larger number of administrators and

** The work by Fabian Suchanek has been partially funded by the European Research Council under the European Community's Seventh Framework Programme (FP7/2007-2013) / ERC grant Webdam, agreement 226513. http://webdam.inria.fr/

services. We argue that, when the complexity of the knowledge represented in a SeCo deployment increases, the SRF model might benefit from adapting ideas from the Semantic Web technologies. Therefore, we propose to substitute the topmost level of the SRF (the service marts) with an ontology. We argue that this will reduce the maintenance effort for a SRF with a large number of data sources, and facilitate the interaction of SeCo with ontology-based systems.

In order to motivate why the SRF model used in SeCo may need to be extended with an ontological representation, we start with a example.

1.1 Motivation

The SRF model leaves room for modeling the same data source in different ways. Assume for example the availability of two search services (Figure 1). The first, ws_1, exposes information about movies played in theaters located close to a given location. The second service, ws_2, queries a repository containing information about movies. The access patterns of these two Web Services can be described on the conceptual level of the SRF as two service marts `MOVIE` and `THEATRE` (see Figure 2), where the attribute `Movie` of the service mart `THEATRE` is linked to the attribute `Title` of the service mart `MOVIE` .

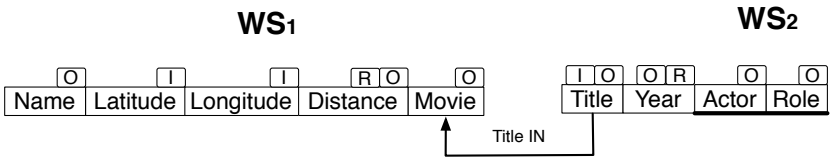

Fig. 1. The Access Patterns for ws_1 and ws_2 on the logical level.

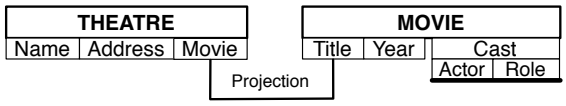

Fig. 2. Two service marts for ws_1 and ws_2 on the conceptual level.

Now let's assume the availability of a service ws_3, which, given a place, searches for all the *actor*s that were born there and returns a description of the actor, including the list of the movies he played in (Figure 3).

As we explained in the introduction, ideally, each service belongs to one Service Mart. The service ws_3 does not belong to any of the service marts that have already been defined. Therefore, the SRF administrator is confronted with two

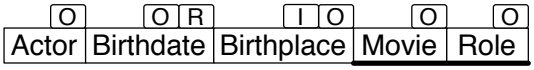

Fig. 3. The Access Pattern for ws_3.

choices: extend the existing `MOVIE` service mart to accommodate the additional parameters brought by ws_3 (as in Figure 4(a)), or create a new `ACTOR` service mart (as in Figure 4(b)). Since structured attributes can replicate the attributes of other concepts, both options are valid and none of them is a priori better than the other.

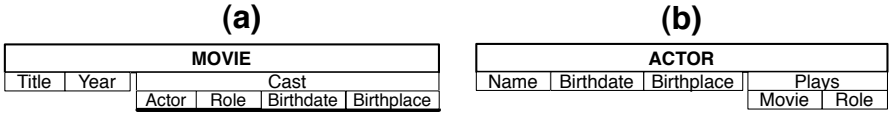

Fig. 4. Examples of alternative evolution of the service marts

If there are multiple SRF administrators, different SRF administrators may choose different (and potentially redundant or incompatible) modeling approaches. In the case of a large scale deployment of a Search Computing system, with potentially hundreds of services and dozens of administrators, these small incompatibilities may quickly add up, leaving the SRF model cluttered with redundancies and inconsistencies. These inconsistencies can lead to major maintenance problems once the size of the SRF surpasses what a single human can assess.

1.2 Contribution

In this chapter, we propose an evolution of the SRF knowledge representation model that will allow it to cope better with some of the maintenance challenges described above. Our proposed model leverages the ontological model that is used in the Semantic Web. It builds on ANGIE [12], a novel approach for Web service description based on the YAGO Ontology [14]. We show that our proposed model not only eases the problem of design alternatives, but also brings additional benefits for the maintenance of the SRF. The new model does not come without challenges of its own. Therefore, this chapter pursues a rather inspirational goal, laying the ground for further investigation.

The chapter is structured as follows: Section 2 discusses the related work; in Section 3, we present YAGO and ANGIE. In Section 4, we propose a merger of the ANGIE knowledge representation model and the SeCo model. In Section 5 we elaborate on the properties of our proposed hybrid model. Finally Section 6 discusses potential future work and concludes.

2 Related Work

The combination of ontologies and services has been proposed before [10,12,13]. The ultimate goal of such an endeavor is to enable a higher degree of automation for the tasks involved in the life-cycle of service based applications. These include (among others) the discovery and selection of services, their composition, their execution and their monitoring. As proposed first in [13], ontologies can be used to model four types of service semantics: *data semantics* (the semantics pertaining to the data contained in the data source), *functional semantics* (the semantics pertaining to the functional capabilities of the service), *non-functional semantics* (the semantics related to the non-functional aspects of the service, such as security or reliability) and *execution semantics* (the semantics related to the invocation, result processing and exception handling of the service).

Several ontology have been proposed to describe these semantics and annotate services. The most well-known ones are OWL-S [2], WSMO [9], SAWSDL [7] and WSMO Lite [16]. Some of them hypothesized the need to semantically describe services at a level of detail that even requires to define a specific ontological language, e.g., the Web Service Modeling Language [6]. All these existing approaches have been perceived unsuited by practitioners in large scale deployments; mostly because of the extra cost of annotating services.

To avoid this pitfall, we decided to look for a light weight approach possibly based on a minimal ontological language. Therefore, we turn our interest to ANGIE [12], a novel approach that is centered on YAGO Ontology [14]. ANGIE leverages the "lightest" of the ontological languages, RDF. This model builds only on typed resources and labeled links. ANGIE uses RDF to describe the input and the output parameters of Web Services and to orchestrate their invocation in order to dynamically extend YAGO with instances loaded from the Web. In the remainder of the chapter we report our initial findings on how ideas from the ANGIE model can ease the definition of new services in SeCo.

3 YAGO and ANGIE

3.1 YAGO

YAGO [14] is a large semantic knowledge base (an *ontology*). It contains 3 millions entities (such as movies, cities, universities and people) and 28 millions facts about them (such as birth dates, appearances in movies, geographical location etc.). YAGO was constructed automatically from Wikipedia and WordNet [8]. In YAGO, knowledge is represented in the RDFS model [4]. This model can be seen as a directed labeled multi-graph, in which nodes represent entities and edges represent relationships between the entities. For example, the node *Apollo Theatre* is linked to the node *Shall we dance* by an edge labeled *shows* (see Figure 5). The labels of the edges are called *relations*. YAGO uses a set of 100 predefined relations. These include a variety of link types, such as *bornIn*, *locatedIn*, *directedMovie* or *isCapitalOf*.

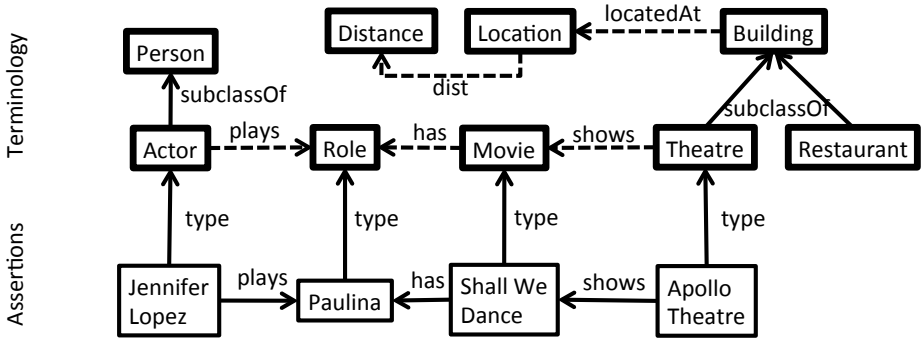

Fig. 5. An excerpt from an RDF ontology.

RDF knowledge bases distinguish between instances (such as *Jennifer Lopez*, drawn thin) and concepts, i.e., groups of similar instances (such as *actor* or *movie*, in bold). Instances and concepts are both nodes in the RDF graph. For example, the concept *movie* is a node in the RDF graph. An instance is linked to its concept by the relation *type*. A concept is linked to a more general concept by the relation *subclassOf*. For example, the sub-concept *actor* is linked to the super-concept *person* in this way. Whenever an instance is an instance of a sub-concept, it is automatically an instance of all of its super-concepts.

Relations are by themselves nodes in the RDF model. This makes it possible to talk about properties of relations in the same way as about instances. For example, to say that the domain of *shows* is the concept *theatre*, we can link the node of *shows* to the node *theatre* by the relation *hasDomain* (not shown in the figure). For illustration, we draw the relations as dashed arrows between the domain concept and the range concept (as shown in Figure 5). Every instance of a sub-concept of *theatre* (e.g., every instance of *3d-theatre*) is also an instance of *theatre*. Thereby, the relation *shows* applies automatically to all instances that live somewhere below the concept *theatre*.

Different from classical database models, RDF models are inherently *schema-less*. This means that any instance can be linked to any other instance with any relation, as long as the domain and the range constraint of the relation are not violated.

3.2 ANGIE

ANGIE [12] is a system that uses Web services to extend YAGO. ANGIE requires the manual registration of Web service and a manual mapping of the input and the output of the services to the concepts of the ontology[3]. Once the Web services

[3] Such a mapping captures in a very light-weight way the *data semantics*, since all data types exchanged with the Web Service are mapped to ontological concepts. It also captures part of the *functional semantics*, since ANGIE can only model Web Services

have been registered, ANGIE allows answering queries on the knowledge base. Whenever the data in the knowledge base is not sufficient to answer a query, ANGIE calls the Web services to retrieve the required additional information. ANGIE can automatically determine the Web services that have to be called to answer the query and it can automatically combine different Web services should that be necessary. This process is transparent to the user, so that the user has the impression of browsing a huge knowledge graph – even though this graph is extended on the fly behind the scenes with the data from the Web services. While ANGIE can deal with arbitrarily-shaped Web services, it cannot deal with ranking of the results.

ANGIE works on *conjunctive SPARQL queries*. These queries can be thought of as RDF graphs that may have variables in the place of the nodes. For example, a user may ask for all movies of the Apollo Theatre by the query depicted in Figure 6. An answer to such a query is a subgraph of the ontology that is isomorphic to the query. In the example, we can match the query on the ontology depicted in Figure 5 with $?m =$*Shall we dance*. Therefore, $?m =$*Shall we dance* is an answer to the query.

Web Service Orchestration If a query cannot be matched on the ontology, or if we want to retrieve more answers than the ontology knows, ANGIE can resort to Web services. A Web service is represented just like a query: as an RDF graph that can contain variables in the place of nodes. Each edge is either an *input edge* or an *output edge*. Figure 6 shows a Web service at the top right, which requires as input (solid) a variable $?a$ that must be an theatre and delivers as output (dashed) a fact that the theatre shows some movie $?b$ and that $?b$ is a movie. Before the Web service can be called, all variables in the input edges have to be instantiated. When the Web service returns its results, these are instantiations of the variables in the output edges.

When ANGIE receives a query, it tries to *cover* the query graph with (1) edges from the ontology or (2) output edges of Web services. Consider the example in Figure 6. We start with the query at the bottom. We cover the query with an instantiation of the Web service (on the layer above). The Web service covers the query edge (solid) with an output edge (dashed). It introduces an additional output edge (that $?m$ is of type *movie*, dashed), which we did not ask for. It also introduces an input edge (that the Apollo Theatre has to be a theatre, solid). Now, the procedure is repeated: Every input edge has to covered again, either by the output edge of another Web service or by an edge from the ontology. In the example, the fact that the Apollo Theatre is a theatre is already in the ontology. Therefore, we can cover this input edge (solid) with the edge from the ontology (top layer, dashed). Now, every input edge and the query is covered. If we call the Web services from top to bottom, we will receive answers to the query. In our example, there is only one Web service, but if there are

that provide information about a give topic or entity, and part of the *execution semantics*, since it describes input and output, but cannot handle exceptions. No formal description of the *non-functional semantics* is provided.

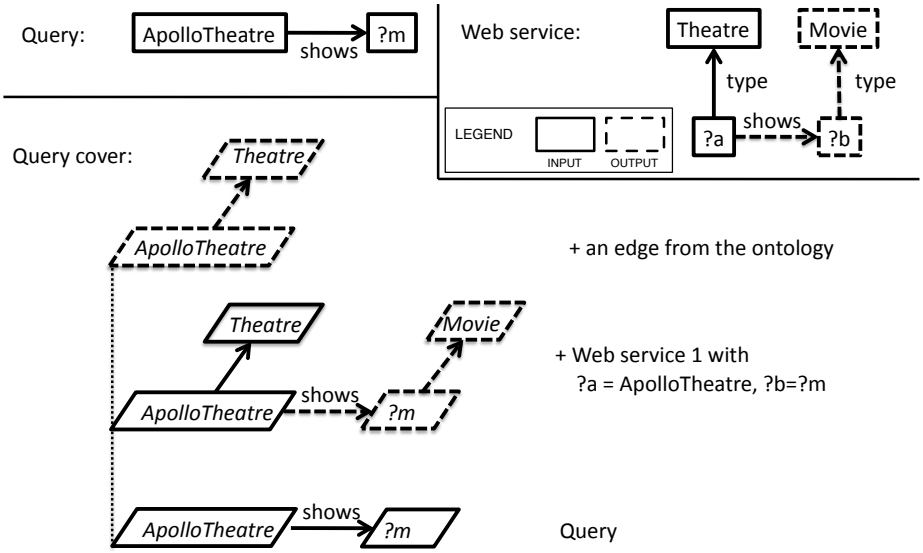

Fig. 6. A user query, a Web service, and a query cover.

multiple Web services, then the outputs of one service are "piped" into the inputs of another service. ANGIE implements a sophisticated scheduling algorithm that can compute such query covers efficiently and give preference to covers that are likely to return more answers. This works even when the query covers are recursive.

All data retrieved from the Web services is added to the ontology. Thereby, future queries can make use of the knowledge that has already been computed.

Virtual Web Services In some cases, the way the ontology models the data and the way the Web services return the data may not coincide. Consider for example the Web services depicted in Figure 7. While the first Web service can return the actors for a given movie, the second service can return not only the actors but also their roles. If the user asks only for the movies (as in the figure), Web service 1 can be applied, but Web service 2 cannot, because none of its output edges matches the query. Even the edges from the ontology cannot be applied, since the ontology models roles as a separate entity.

To bridge such mismatches, ANGIE supports *virtual services*. A virtual service is a pseudo Web service that does not have a physical Internet service behind it. "Calling a virtual service" means matching the input edges and then adding the output edges to the query cover – without any physical Web service call. In the example in Figure 7, the virtual service allows adding a *playsIn* fact, if there are *plays* and *has* facts for a role.

With the virtual service, our query can be answered even with Web service 2: We first call Web service 2 to retrieve movies together with their roles. Then,

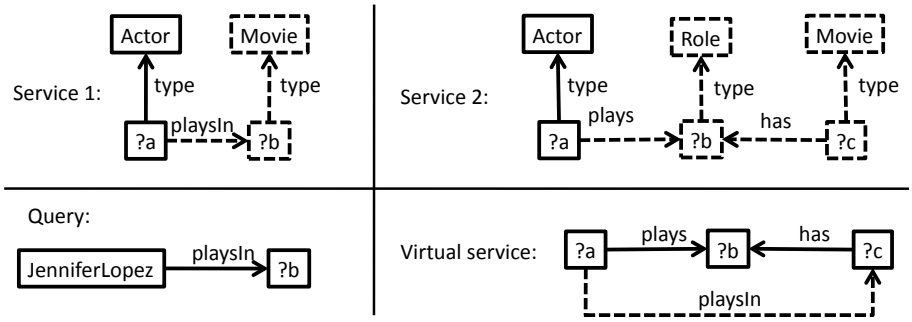

Fig. 7. A virtual Web service.

we "call" the virtual Web service to transform the output into *playsIn* facts that answer the query. Such a call does not involve any physical Web service call. It is a pure data manipulation on the query edges, which allows us to bridge data modelizations of different granularity.

4 Toward an ontological representation of the access patterns

Inspired by the ANGIE approach, we propose to substitute the Service Mart layer of the SRF model with the YAGO ontology. The role of Service Marts will be taken over by the concepts and relations of the ontology – just like explained in Section 3.2. Thereby, the conceptual layer of the SRF for the running example will be the ontology depicted in Figure 5.

In addition, we propose to describe the access patterns in the way Web Services are described in ANGIE: Each access pattern becomes a graph. The edges are labeled with relationships from the ontology. The nodes can be either constants from the ontology or variables. Each attribute of an access pattern becomes an edge that is labeled with the corresponding relationship from the ontology. This edge connects to a node with a variable. Structured attributes become star-shaped patterns. As in ANGIE, nodes are labeled as *input nodes* or *output nodes*. The type of a variable node is indicated by an outgoing `type` edge to a concept node.

To cope with the notion of *ranking* (typical for SeCo), ANGIE's representation of Web services has to be extended so as to allow not only input nodes and output nodes, but also *ranking nodes*. In every access pattern graph, there can be at most one node that is labeled as a ranking node. Such a node has to be an output node. In the illustrations, we represent a ranking node by a filled dashed frame. Figure 8 shows how the access patterns ws_1 and ws_2 from the introduction can be modeled; ws_1, which belonged to the Service Mart *Theater*, requires as input (solid) a variable ?a, a location, and produces as output (dashed) the theaters ?c, located at a distance ?b from the input location, and showing some

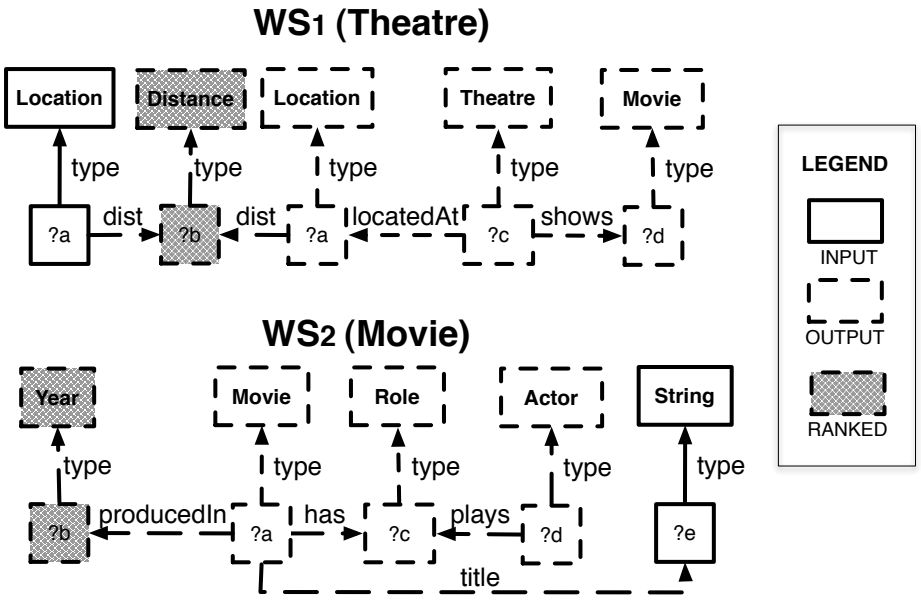

Fig. 8. RDF representation of two Access Patterns shown in Figure 3.

movie ?d; the results of ws_1 are also ordered according to the distances ?b. Likewise, ws_2, which belonged to the Service Mart *Movie*, requires as input a the title ?e of movie ?a, and it produces as output the actors ?d that acted in the movies with the role ?c; results are ordered according to the production year ?b.

Noteworthily, the novel RDF representation is isomorphic w.r.t. the original one. This assures compatibility with the service description exploited by the other levels of the architecture. Indeed, we remark that all of our proposed changes happen purely at the level of service modeling. No changes to the Web service composition algorithms or the ranking algorithms of SeCo are necessary.

5 Properties of the Proposed Model

We note that the proposed representation helps in coping with the issues mentioned in the introduction: as Figure 9 depicts, there is no ambiguity whether the information brought by ws_3 shall be stored within the service mart `MOVIE` or `ACTOR`. The administrator just has to describe the access pattern as a query on YAGO ontology: the access pattern selects instances of actors (`?a type Actor`) based on their birthplace (`?a bornIn ?b`), ranked by the date of birth (`?a bornWhen ?e`) and lists the movies they played in (`?d type Movie; ?a plays ?c; ?d has ?c`).

Given that every concept and property exists exactly once, there are fewer possibilities for inconsistencies to appear in the ontological model.

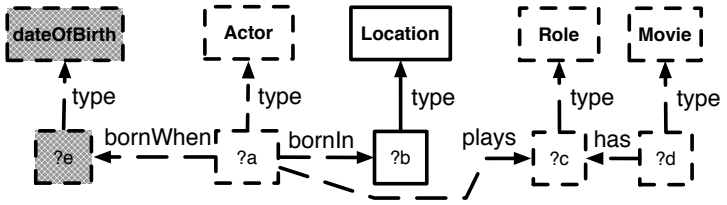

Fig. 9. The Web service ws_3 in the new model.

In addition to the specific maintenance case that we developed through the paper, the use of an ontological description for the conceptual level brings several advantages: notably, the ontology can evolve independently from the registered data sources. In the SRF model that we are proposing, the ontology models the world and the access patterns model the Web Services. An extension of the ontology will not influence the access patterns. For instance, the subclass `Restaurant` of `Building` can exist in the ontology even if no access pattern refers to it. Vice versa, the addition or removal of an access pattern will not influence the ontology (provided that all necessary relations and concepts are present). Thereby, the roles of the ontology and the access patterns are clearly defined and distinct.

Since the ontology allows for the creation of sub-concepts, more specific concepts can be created without having to redo the work that has been done for the super-concept. The attributes of the sub-concept are automatically consistent with the attributes of the super-concept. By design, a Web service that delivers an instance of the sub-concept can also be used to deliver an instance of the super-concept. Furthermore, through the domains and ranges of relations, the target type of a relation is explicitly defined. There is no need to replicate this information with every link or every Web service. Target types are an inherent part of the model.

Since relations are first-class citizens of the model, the joinability of attributes follows directly from the definition of the relations and the access patterns. For example, if a Web service returns movies, then the joinability with another Web service that returns movies follows from the fact that both output variables are of type *movie*. This also works across different levels in the concept hierarchy: If one Web service returns 3d-movies and the other Web service returns silent movies, these Web services are still joinable, since both concepts are sub-concepts of *movie*. By factoring out this common information from the level of Web services to the level of classes, the ontological model avoids redundancy.

The only maintenance task left to SRM administrators is the extension of the ontology when the registration of a new Web Service requires adding new concepts and relationships to YAGO. For instance, if the relationship `bornIn` is not present in the ontology, but is brought in by a Web service, then this relationship has to be added by the SRF administrator. This is not a trivial task per se, but at least we can rely on established methodological approaches

(e.g., METHONTOLOGY [5], On-To-Knowledge [15], and DILIGENT [11]) for ontology development and maintenance.

6 Conclusions and Future Work

In this article, we have discussed some of the maintenance challenges that SeCo will face when more users, services and administrators start using the system in parallel. We have argued that, for this task, the SeCo knowledge representation model would benefit from a more ontological design. We have proposed and studied an ontological adaptation of the SeCo model, inspired by the model used in the ANGIE system. We have shown that the new model mitigates some of the maintenance challenges, thus contributing to SeCo's fitness for going mainstream.

We believe that, apart from easing some of the maintenance challenges, an ontological top layer of SeCo opens the door to a wide range of possible interactions between SeCo and existing ontologies. These include not only YAGO but also the vast resources of the Linking Open Data Project (LOD) [1]. Whilst the LOD has not been the focus of this article, we are confident that the shared knowledge representation will ease knowledge exchange between the LOD and SeCo in the future – e.g., by answering queries with data from the LOD in case a Web service is not available.

We believe that there is much further research potential in the combination of ANGIE and SeCo. For example, the orchestration algorithm of ANGIE can only combine the outputs of one Web Service with the inputs of another Web Service (i.e., it can only perform *pipe joins*, in SeCo terminology). Therefore, some registered Web Service cannot be used directly. SeCo, in contrast, can master the combinatory explosion that appears when results from multiple Web Services are combined, because it uses rank aware parallel join operators.

On the other hand, ANGIE is very good at answering SPARQL queries. This is an avenue on which SeCo could benefit. One possibility to combine the advantages of SeCo and ANGIE would be to register Web Services both in ANGIE and in SeCo. Whenever a user issues a SPARQL query on YAGO, ANGIE could orchestrate (virtual and real) Web Services normally whenever it can pipe the output of one Web Service into the input of another Web Service. When ANGIE faces the problem to perform parallel joins on the results of multiple Web Services, it could delegate the execution of this part of the orchestration to SeCo. This is just one example for the research possibilities that wait to be discovered in further cross-fertilization of these approaches.

References

1. C. Bizer, T. Heath, and T. Berners-Lee. Linked data - the story so far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009.
2. M. Burstein, J. Hobbs, O. Lassila, D. Mcdermott, S. Mcilraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, and K. Sycara. Owl-s: Semantic markup for web services. Website, 2004.

3. S. Ceri, editor. *Search Computing*, volume 5950 of *Lecture Notes in Computer Science*. Springer, 2010.
4. W. W. W. Consortium. Rdf primer, w3c recommendation. http://www.w3.org/TR/rdf-primer/, 2004.
5. Ó. Corcho, M. Fernández-López, A. Gómez-Pérez, and A. López-Cima. Building legal ontologies with methontology and webode. In V. R. Benjamins, P. Casanovas, J. Breuker, and A. Gangemi, editors, *Law and the Semantic Web*, volume 3369, pages 142–157, 2003.
6. J. de Bruijn, D. Fensel, M. Kerrigan, U. Keller, H. Lausen, and J. Scicluna. *Modeling Semantic Web Services: The Web Service Modeling Language*. Springer, Berlin, 2008.
7. J. Farrell and H. Lausen. Semantic annotations for wsdl and xml schema, August 2007.
8. C. Fellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.
9. D. Fensel, A. P. Holger Lausen, J. de Bruijn, M. Stollberg, D. Roman, and J. Domingue. *Enabling Semantic Web Services: The Web Service Modeling Ontology*. Springer, Berlin, 2006.
10. S. A. McIlraith, T. C. Son, and H. Zeng. Semantic web services. *IEEE Intelligent Systems*, 16(2):46–53, 2001.
11. H. S. Pinto, S. Staab, and C. Tempich. Diligent: Towards a fine-grained methodology for distributed, loosely-controlled and evolving engineering of ontologies. In R. L. de Mántaras and L. Saitta, editors, *ECAI*, pages 393–397. IOS Press, 2004.
12. N. Preda, G. Kasneci, F. Suchanek, T. Neumann, and W. Yuan. Active knowledge: Dynamically enriching rdf knowledge bases by web services (angie). In *International Conference on Management of Data (SIGMOD 2010)*. ACM, 2010.
13. K. Sivashanmugam, K. Verma, A. P. Sheth, and J. A. Miller. Adding semantics to web services standards. In L.-J. Zhang, editor, *ICWS*, pages 395–401. CSREA Press, 2003.
14. F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A Core of Semantic Knowledge. In *16th international World Wide Web conference (WWW 2007)*, New York, NY, USA, 2007. ACM Press.
15. Y. Sure, S. Staab, and R. Studer. Methodology for development and employment of ontology based knowledge management applications. *SIGMOD Record*, 31(4):18–23, 2002.
16. T. Vitvar, J. Kopecký, J. Viskova, and D. Fensel. Wsmo-lite annotations for web services. In S. Bechhofer, M. Hauswirth, J. Hoffmann, and M. Koubarakis, editors, *ESWC*, volume 5021 of *Lecture Notes in Computer Science*, pages 674–689. Springer, 2008.